**Chapter 5**

**5.10 What is the difference between the following LC-3 instructions A and B? How are they similar? How are they different?**

**A: 0000 1 1 1 101010101**

**B: 0100 1 11101010101**

A is a Conditional Branch and in this question A is also an Unconditional Branch which have the similar function to B that they can jump to the location specified by adding the sign-extended PCoffset9/11 field to the incremented PC. However, the range of PC is different. A is -256~255, while B is -1024~1023.

**5.15 State the contents of Rl, R2, R3, and R4 after the program starting at location x3100 halts.**

**Address Data**

**0011 0001 0000 0000 1110 001 000100000**

;x3100, LEA R1,0x20, R1🡨x3121

**0011 0001 0000 0001 0010 010 000100000**

;x3101, LD,0x20, R2🡨men[x3122]= 0100 0101 0110 0110

**0011 0001 0000 0010 1010 011 000100000**

;x3102, LDI, R3, 0x20

;R3🡨men[men[x3123]]=men[0100 0101 0110 0111]=1010 1011 1100 1101

**0011 0001 0000 0011 0110 100 010 000001**

;x3103, LDR R4,R2, 0x1, R4🡨men[R2+#1]=1010 1011 1100 1101

**0011 0001 0000 0100 1111 0000 0010 0101** ;x3104, TRAP 0x25, halt

**0011 0001 0010 0010 0100 0101 0110 0110** ;x3122

**0011 0001 0010 0011 01000101 01100111** ;x3123

**0100 0101 0110 0111 1010 1011 1100 1101** ;x4567

**0100 0101 0110 1000 1111 1110 1101 0011** ;x4568

R1:0x3121

R2:0100 0101 0110 0110 0x4566

R3:1010 1011 1100 1101 0xabcd

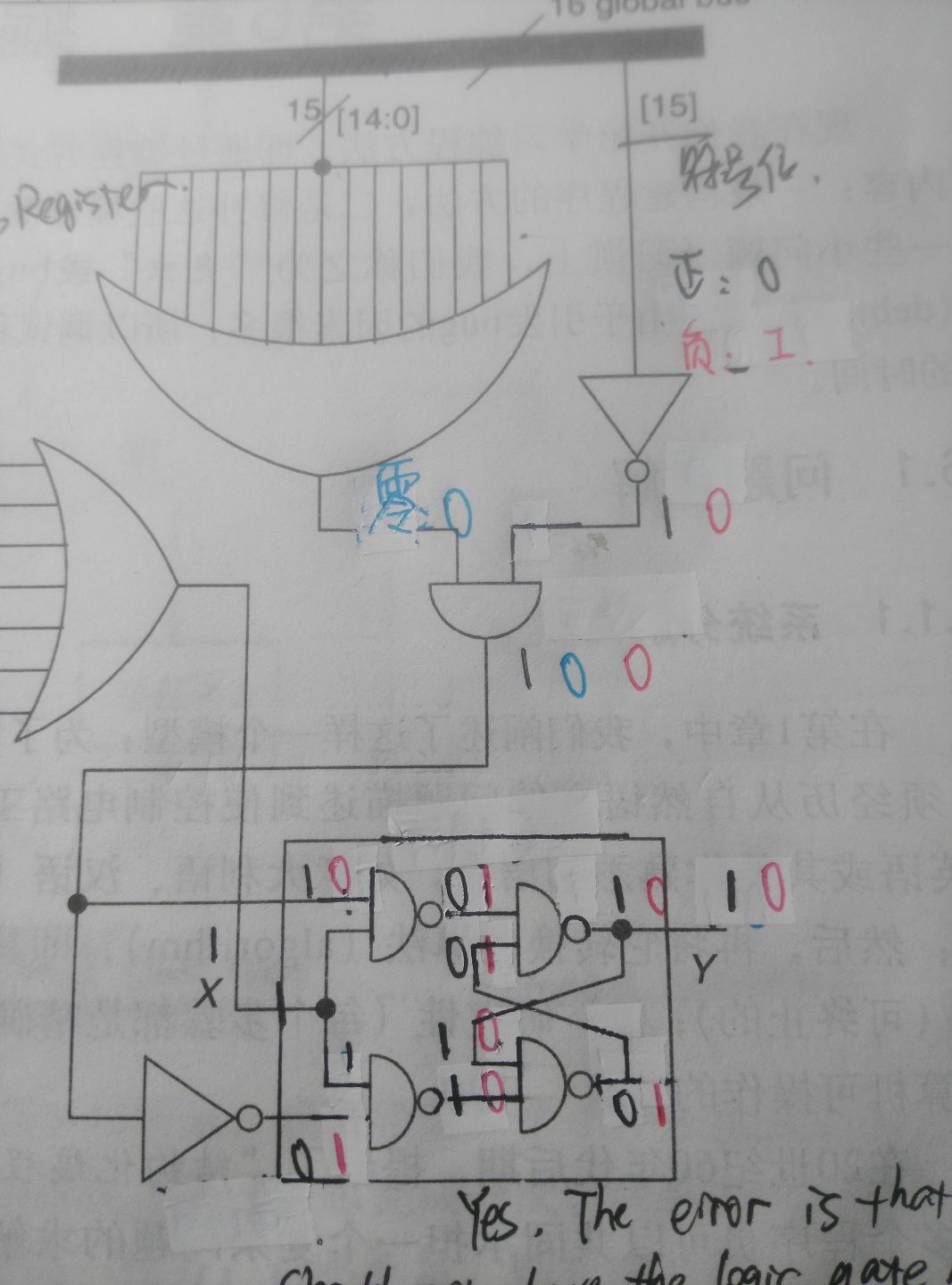
R4:1111 1110 1101 0011 0xabcd

**5.41 A part of the implementation of the LC-3 architecture is shown on the top of the next page.**

***a.* What information does Y provide?**

***b.* The signal X is the control signal that gates the gated D latch. Is there an error in the logic that produces X?**

a.



Y is a Positive Condition Code.

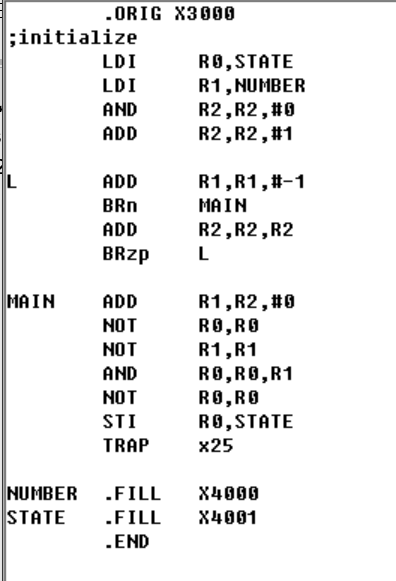
|  |  |  |
| --- | --- | --- |
| P | 0 | N |
| 1 | 0 | 0 |

b. Yes. There is an error that A should not be there because IR needed there is some code that can change the condition of the registers, while X is a constant number 1 for the BR opcode.

**Chapter 6**

**6.3 Recall the machine busy example from previous chapters. Suppose memory location x4000 contains an integer between 0 and 15 identifying a particular machine that has just become busy. Suppose further that the value in memory location x4001 tells which machines are busy and which machines are idle. Write an LC-3 machine language program that sets the appropriate bit in x4001 indicating that the machine in**

**x4000 is busy. For example, if x4000 contains x0005 and x4001 contains x3101 at the start of execution, x4001 should contain x3121 after your program terminates.**



**.ORIG X3000**

**;initialize**

**LDI R0,STATE ;load the value of STATE to R0**

**LDI R1,NUMBER ;load the value of NUMBER to R1**

**AND R2,R2,#0 ;initialize R2**

**ADD R2,R2,#1 ;start from 0000 0000 0000 0001**

**L ADD R1,R1,#-1 ;counter**

**BRn MAIN**

**ADD R2,R2,R2 ;move 1 to left**

**BRzp L**

**MAIN ADD R1,R2,#0**

**NOT R0,R0 ;NOT[NOT[R0] AND NOT[R1]]=OR**

**NOT R1,R1**

**AND R0,R0,R1**

**NOT R0,R0**

**STI R0,STATE ;store the value of state from R0**

**TRAP x25 ;halt**

**NUMBER .FILL X4000**

**STATE .FILL X4001**

**.END**

**6.9 Using the iteration construct, write an LC-3 machine language routine that displays exactly 100 Zs on the screen.**

![C:\Users\laylalaisy\AppData\Roaming\Tencent\Users\516278695\QQ\WinTemp\RichOle\`M[@WRN2MJCKL{](ENUD@NK.png](data:image/png;base64,)

**.ORIG x3000**

**LD R0,ASCII ;load the value of the ASCII to R0**

**LD R1,COUNT ;load the total times need to perform**

**OUTPUT TRAP x21 ;output**

**ADD R1,R1,#-1 ;counter**

**BRp OUTPUT**

**TRAP X25 ;halt**

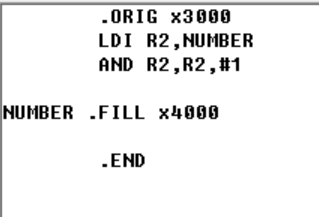
**ASCII .FILL x005A**

**COUNT .FILL #100**

**.END**

**6.10 Using the conditional construct, write an LC-3 machine language routine that determines if a number stored in R2 is odd.**

First store the original number in the address of x4000. Then execute the program, if the value of R2 is 1, then the original number is odd. If the value of R2 is 0, then the original number is not odd.



**.ORIG x3000**

**LDI R2,NUMBER ;load the value from x4000**

**AND R2,R2,#1**

**NUMBER .FILL x4000**

**.END**